Dynamic Partitioning for Concurrent Waveform Relaxation-Based Circuit Simulation

Lena Peterson,
Physics of Computation Laboratory,
California Institute of Technology,
MS 256-80, Pasadena, CA, 91125, USA

Sven Mattisson,
Department of Applied Electronics,
Lund University,
P.O. Box 118, S-22100 Lund, Sweden

Abstract–We present a new dynamic circuit partitioning algorithm for the waveform relaxation method. Such an algorithm dynamically changes the partitioning as the simulation proceeds through the simulation interval. The proposed algorithm is suitable for implementation on a multicomputer. Experimental results show that the algorithm decreases the runtimes for circuits where a good static partitioning is hard to find. This is true both in the ideal case, that is when communication overhead for the repartitioning is not included and the load is distributed as evenly as possible among the processors, and in the real-world case, when all the partitioning overhead and load imbalance is included in actual measured run times.

I. Introduction

The purpose of this project was to investigate the usefulness of dynamic circuit partitioning algorithms for a waveform relaxation-based circuit simulator designed for multicomputers. A multicomputer is a concurrent message-passing computer with local memory space for each processing node. For such a computer there is an important tradeoff between the irregularity in the partitioning and the achievable parallelism. A dynamic partitioning algorithm changes the circuit partitioning as the simulation proceeds through the simulation interval. Thus, ideally, a highly irregular partitioning will be used only when the current circuit state indicates so.

Waveform relaxation is an iterative method which can be used for performing transient analysis. That is, it is a method which solves the equations which we can form from the description of a circuit, typically by applying Kirchhoffs current law to the circuit. These equations, representing the circuit dynamics, is a system of DAEs (differential-algebraic equations). Such a system can be partitioned into a number of subsystems (or blocks) each containing at least one DAE. Using the waveform relaxation method, we can then solve these blocks separately. When solving one block all the other blocks are *relaxed*, that is, the solutions from the other blocks are assumed to be static. In this manner it is possible to solve for the unknowns in one block over a time interval (called a *time window*) without interaction with the other blocks. The partitioning of the DAE system is extremely important for the convergence speed for the iterative WR method. Any iteration scheme can be used for the global iterations over the blocks, for example Gauss-Seidel or Jacobi iteration methods. In this investigation we concentrate on the Jacobi iteration method since it gives the best asymptotic speedup [5].

II. Dynamic Partitioning Criteria

The goal of the partitioning methods for the system of DAEs, is to merge equations that are strongly coupled so that the resulting subsystems are only loosely coupled. The proposed dynamic partitioning method computes the couplings from the Jacobian matrix **J** which contains the partial derivatives of the model equations at the current state. An efficient dynamic partitioning method has to make its repartitioning decisions based on the information that can be obtained from the current partitioning. Thus, our dynamic partitioning method consists of two different parts, one *division criterion* which is used to decide if currently clustered nodes should be separated and one *merging criterion* which is used to decide if current blocks should be merged.

[1]The research described in this paper was supported in part by NUTEK Dnr 9001159 and in part by the Defense Advanced Research Projects Agency, DARPA Order number 6202, monitored by the Office of Naval Research under Contract Number N00014-87-K-0745. This research is described in detail in Peterson's PhD thesis [1].

A. The Division Criterion

The division criterion treats each diagonal block Jacobian matrix as if it were a Jacobian matrix for a self-contained equation system. The value of the return voltage transfer ratio, T_{mn}, between circuit nodes m and n belonging to the block j, determines if they are still to belong to the same block. The return voltage transfer ratio is defined as the "roundtrip" voltage gain:

$$T_{mn} \equiv T_{nm} = A_m^n \cdot A_n^m. \tag{1}$$

In our division criterion the return voltage transfer ratio is computed from the inverse of the diagonal block Jacobian, $\mathbf{J}_{jj}$ for block j, as

$$T_{mn} = \left| \frac{(\mathbf{J}_{jj}^{-1})_{mn}(\mathbf{J}_{jj}^{-1})_{nm}}{(\mathbf{J}_{jj}^{-1})_{mm}(\mathbf{J}_{jj}^{-1})_{nn}} \right| \tag{2}$$

Since the blocks are assumed not to be extremely large, hopefully 1 – 20 circuit nodes, the cost for computing the inverse $\mathbf{J}_{jj}^{-1}$ is not prohibitive even though the calculation is $\mathcal{O}(n^3)$ where n is the size of the block. Furthermore, the inverses of the diagonal Jacobians can be reused when calculating the merge criterion. The division criterion for nodes m and n is

$$T_{mn} > \gamma_d \tag{3}$$

where γ_d is the division limit, which in our experiments has been kept constant throughout each simulation run. It is noteworthy that even though (3) is called the *division* criterion, it is fulfilled when two circuit nodes are *not* to be divided. We obtain the "global" partitioning for the block by finding the components of an undirected graph representing the block. The vertices of this graph correspond to the circuit nodes in the block. A pair of circuit nodes fulfilling (3), has their corresponding vertices connected by an edge. In our current implementation the division criterion is computed locally for each block for each time point in the window iterations during which the repartitioning is computed.

B. The Merging Criterion

The merging criterion is based on the *collapsed iteration matrix* for the block Jacobi method. The collapsed iteration matrix is derived using p-norm inequalities for each row equation. For a blocked system with N blocks it is:

$$\mathbf{T} = \begin{bmatrix} 0 & \left\|\mathbf{J}_{11}^{-1}\mathbf{J}_{12}\right\|_p & \cdots & \left\|\mathbf{J}_{11}^{-1}\mathbf{J}_{1N}\right\|_p \\ \left\|\mathbf{J}_{22}^{-1}\mathbf{J}_{21}\right\|_p & 0 & \cdots & \left\|\mathbf{J}_{22}^{-1}\mathbf{J}_{2N}\right\|_p \\ \vdots & \vdots & \ddots & \vdots \\ \left\|\mathbf{J}_{NN}^{-1}\mathbf{J}_{N1}\right\|_p & \left\|\mathbf{J}_{NN}^{-1}\mathbf{J}_{N2}\right\|_p & \cdots & 0 \end{bmatrix}. \tag{4}$$

This matrix can be viewed as derived from a system matrix. Since the iteration matrices derived from all system matrices differ only by a scalar factor we choose the system matrix that is easiest to compute, that is is the one where the diagonal contains only ones, or when $\mathbf{D} = \mathbf{I}$. We call this system matrix $\mathbf{P}$. It is calculated as

$$\mathbf{P} = -\mathbf{T} + \mathbf{I}. \tag{5}$$

The matrix $\mathbf{P}$ is the one from which we derive the merging criterion. We use the diagonally dominant loop criterion introduced by White [6, 7]. The criterion, applied to the entries of $\mathbf{P}$, is

$$\frac{p_{jk}p_{kj}}{p_{jj}p_{kk}} = p_{jk}p_{kj} > \gamma_m, \tag{6}$$

where γ_m is the merging limit, usually fixed but similar to the division limit it may be changed if desired.

The diagonally dominant loop criterion is a local criterion, and it can, thus, not detect loops of unidirectional couplings among more than two blocks. It would be desirable to use a criterion right-hand side similar to the one of the division criterion (eq:transferratio) also for the merging. That is, we would like to use the inverse of $\mathbf{P}$. In practice, however, it would be extremely expensive to invert $\mathbf{P}$ since this matrix does not exist physically anywhere in the database. Furthermore, for each waveform iteration, each off-diagonal element of $\mathbf{P}$, p_{jk}, is a waveform of values calculated for the time points chosen by the integration algorithm for block j. In order to invert $\mathbf{P}$ for a number of time points over the present time window each p_{jk} must be interpolated to each desired time point and then the matrix can be built somewhere to be inverted (possibly via a distributed inversion algorithm).

III. Criteria Calculations in Practice

The program used for the experiments, called CONCISE [2], is a circuit simulator for transient analysis of MOS circuits. It is written in C and uses the Cosmic Environment/Reactive Kernel message-passing primitives [3, 4]. These primitives support the programming model where the computational unit is the process and each process has its own local memory-space. A process is an instance of a program that causes messages to be sent and received. CONCISE consists of a number of *solver* processes, each responsible for computing the solution to a number of blocks for each window iteration and one *scheduler* process which collects convergence data from the solvers and decides when a window has converged or when a time window has to be split.

The inversion of the diagonal Jacobian matrices is the core computation in both the division and the merging criteria. The matrix inversion is performed using the already LU-factorized Jacobian matrix from the last Newton iteration in the current time point. If both criteria are calculated during the same iteration (which is not necessary) the same matrix inversion is used for both of them.

The calculation of the division criterion is performed entirely within one solver process. Communication with other processes is, in principle, only required to decide in what solver processes the new blocks should be placed. In our implementation the merging overrides the division and so communication is also needed to verify that the block is not being merged.

For the division calculation the block is represented as an adjacency matrix **N**, a straightforward matrix representation of the connections in the undirected graph describing the block. Each off-diagonal entry can be either one or zero. A non-zero entry n_{ij} indicates that circuit node i and j are coupled. All entries in **N** are set to zero at the start of the window iteration during which the coupling calculation is to take place. For each time point, entries are added to the **N** for each pair of circuit nodes that fulfill the division criterion (3). If the graph is connected then the block can not be divided. At each time point during the coupling calculation **N** is checked to see if the block graph is fully connected. If it is no more division calculations have to be performed for the block during the current window iteration.

For the merging calculations communication between solver processes is necessary since the bidirectional coupling between two blocks has to be computed from the entries in the system matrix **P**, derived from the collapsed iteration matrix. The off-diagonal entry p_{ij}, which describes the impact of block i on block j during a time window, becomes a waveform sampled at the same time points as the voltage waveforms. We call such a waveform a *coupling waveform*. It is not feasible to store or communicate all the time points in the coupling waveforms because that would require too much memory. To alleviate these memory requirements, we use a compressed representation for the coupling waveforms. This representation comprises three coupling values, the first, the last, and the middle one of the waveform, plus the largest (*i.e.* the worst) coupling value and its corresponding time point. Thus, only five real numbers are used for storing the information. Additional memory is needed during the calculation of the waveform, but it does not have to be retained once the solution of the particular block is concluded.

IV. Experiments

The circuits used in the experiments are all digital MOS circuits. All in all there are 14 circuits ranging in size from 18 to 1944 circuit nodes (40 to 3348 transistors). All fourteen circuits come from research chips designed either in Lund or at Caltech. The netlists for all the circuits were extracted from the chip layouts.

We tested four different schemes for using the two repartitioning criteria described above. In all four schemes the criteria are computed in a fixed window iteration for each time window. In three of the schemes the division and the merging criteria are both computed in the same window iteration, that is, in window iteration 1, 2, or 3 respectively. The fourth scheme is called the concurrency-preferring partitioning (CPP) since it prefers having many small blocks and slower convergence over few blocks and faster convergence. This is achieved by computing the division as soon as possible, in the first window iteration, while postponing the merging until window iteration 9.

For comparison, we have also used a precomputed partitioning the off-line dynamic partitioning (ODP). In this case the dynamic partitioning is computed from the Jacobians collected form precious simulations of the same circuit using an incremental-time method.

The on-line dynamic partitioning, that is when the repartitioning is computed from data from the on-going run, has been analyzed in two types of experiments. In the first type of experiments, the ideal placement experiments, we disregard the repartitioning overhead and calculate the best possible load balance. Thus, these experiments reflect the partitioning criteria and the partitioning schemes themselves without taking the shortcomings of our coding and the type of computer we used into account. In the second type of experiment we just measure the actual runtime with all existing overhead.

The four dynamic partitioning schemes are compared to two static partitioning methods: the single-equation (SE) partitioning method where each circuit node forms one block and the gate (G) partitioning where circuit nodes connected by the source and drain by the same transistor are clustered.

Plots for the circuit where the dynamic partitioning is most successful, **adder**, are shown in Figures 1 to 3. Note that the improvement in runtime with partitioning limit is not monotonically decreasing when the partitioning limit even for the ODP method.

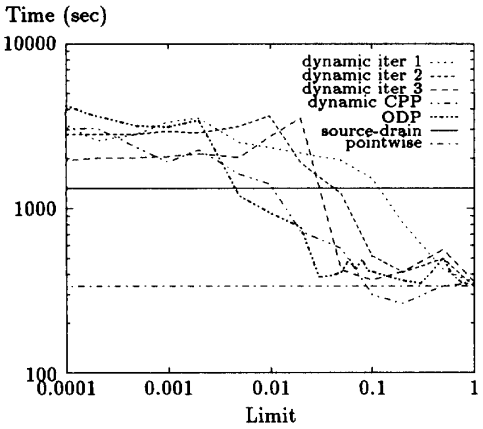

Fig. 1: **Adder**, dynamic on-line partitioning in 32 processors, ideal placement.

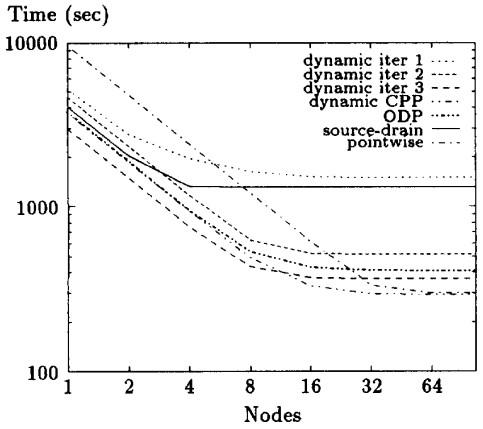

Fig. 2: **Adder**, with division and merging limit = 0.1, ideal placement.

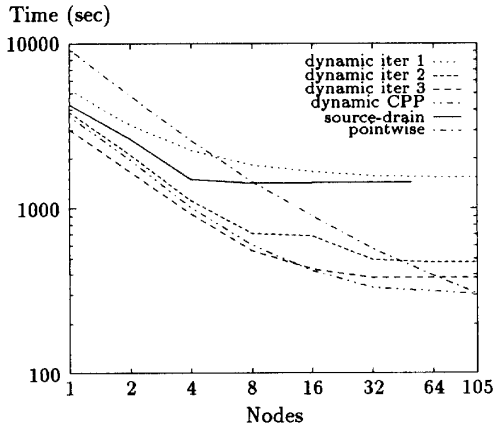

Fig. 3: **Adder** on-line dynamic partitioning, measured run times with division and merging limit = 0.1.

V. Conclusions

In this study we have derived two new dynamic partitioning criteria for the waveform relaxation method, one for the merging of existing blocks and one for the division of existing blocks. The division criterion takes both bidirectional couplings and loops of unidirectional couplings into account. The merging criterion considers only bidirectional couplings. The experiments show that speedup improvement up to an order of magnitude over the static heuristic gate partitioning can be obtained for MOS circuits for which the gate partitioning yields large blocks. For circuits for which the gate partitioning produces only small blocks there is, for most circuits, no improvement.

Acknowledgements

We thank professor Charles L. Seitz for continuous encouragement and innumerable hours of multicomputer time.

References

[1] L. Peterson. *Dynamic Circuit Partitioning for Concurrent Waveform Relaxation-Based Circuit Simulation.* PhD thesis, Dept of Applied Electronics, Lund University, Lund, 1992.

[2] L. Peterson and S. Mattisson. The design and implementation of a circuit simulation program for multicomputers. *IEEE Transactions of Computer-Aided Design of Integrated Circuits and Systems*, March, 1993. In press.

[3] C. L. Seitz, J. Seizovic, and W-K. Su. The C Programmer's Abbreviated Guide to Multicomputer Programming. Technical report Caltech-CS-TR-88-1, Computer Science Dept, California Institute of Technology, Pasadena, 1988. Revised May 1989.

[4] J. Seizovic. The reactive kernel. Technical Report SC-TR-88-10, Computer Science Dept, California Institute of Technology, 1988. MS Thesis.

[5] D. W. Smart. *Parallel Processing Techniques for the Simulation of MOS VLSI Circuits Using Waveform Relaxation.* PhD thesis, Dept of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, 1988.

[6] J White and A.L. Sangiovanni-Vincentelli. Partitioning algorithms and parallel implementations of waveform relaxation algorithms for circuit simulation. In *Proceedings of IEEE International Symposium on Circuit and Systems*, pages 221–224, 1985.

[7] J. K. White and A. Sangiovanni-Vincentelli. *Relaxation Techniques for the Simulation of VLSI Circuits.* Kluwer Academic Publishers, Boston, Massachusetts, 1987.